Конструирование информационных систем на основе интероперабельных сред информационных ресурсов

Технологии промежуточного слоя

- Промежуточный слой расположен между операционной системой и прикладными системами
- Обеспечивает техническую возможность конструирования распределенных, интероперабельных ИС
- Позволяет накапливать репозитории компонентов для их дальнейшего использования при создании новых ИС
- Виды промежуточного слоя
  - Транзакционно-ориентированные
  - Ориентированные на сообщения
  - Объектно-ориентированные
    - CORBA, Java RMI, .NET
  - XML-ориентированные
    - Web Services

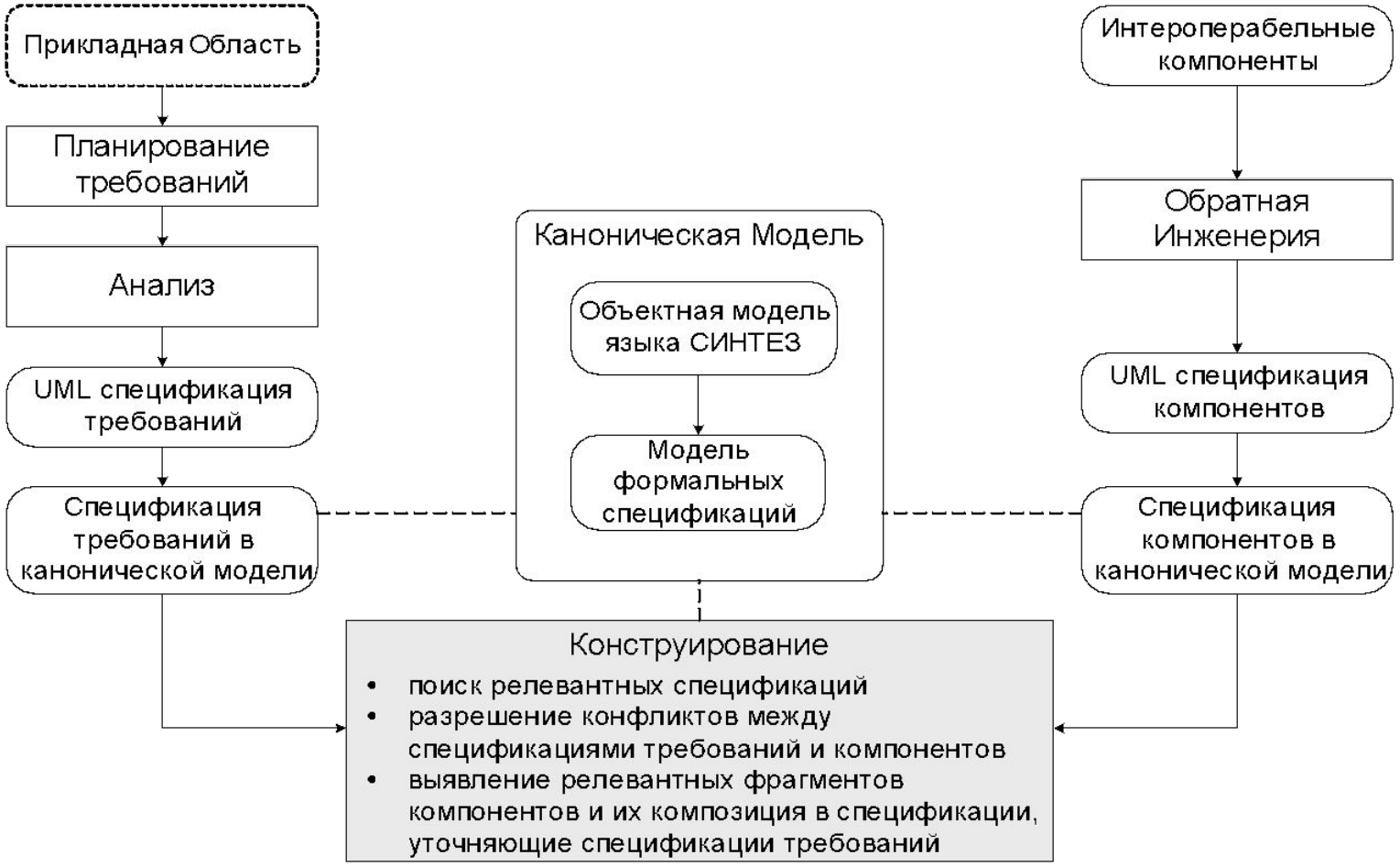
Средства проектирования ИС из компонентов

- Методы и средства компонентно-базированного проектирования ИС
  - Microsoft's Visual Studio
  - Borland's JBuilder
- Основные ограничения
  - не позволяют автоматически устранять различия в спецификациях накопленных компонентов и в спецификациях требований;
  - не направлены на создание корректных композиций компонентов и их фрагментов, уточняющих спецификации требований;
  - не обеспечивают масштабируемого обнаружения компонентов, семантически соответствующих требуемому применению;
  - являются ориентированными на конкретную архитектуру промежуточного слоя.

Композиционный подход

- Композиционный подход к проектированию ИС из компонентов, технически интероперабельных в рамках некоторого промежуточного слоя
- Необходимо решение следующих задач:
  - разработка алгоритмов поиска компонентов, онтологически релевантных спецификации требований;
  - разработка алгоритмов разрешения конфликтов между спецификациями требований и компонентов;
  - разработка алгоритмов выявления фрагментов спецификации компонентов, которые могли бы служить уточнением соответствующих фрагментов спецификации требований;
  - разработка алгоритмов построения композиции таких фрагментов в спецификацию, уточняющую спецификацию требований;
  - создание инструментария эксперта-конструктора ИС на основе перечисленных алгоритмов.

Схема композиционного подхода к проектированию ИС



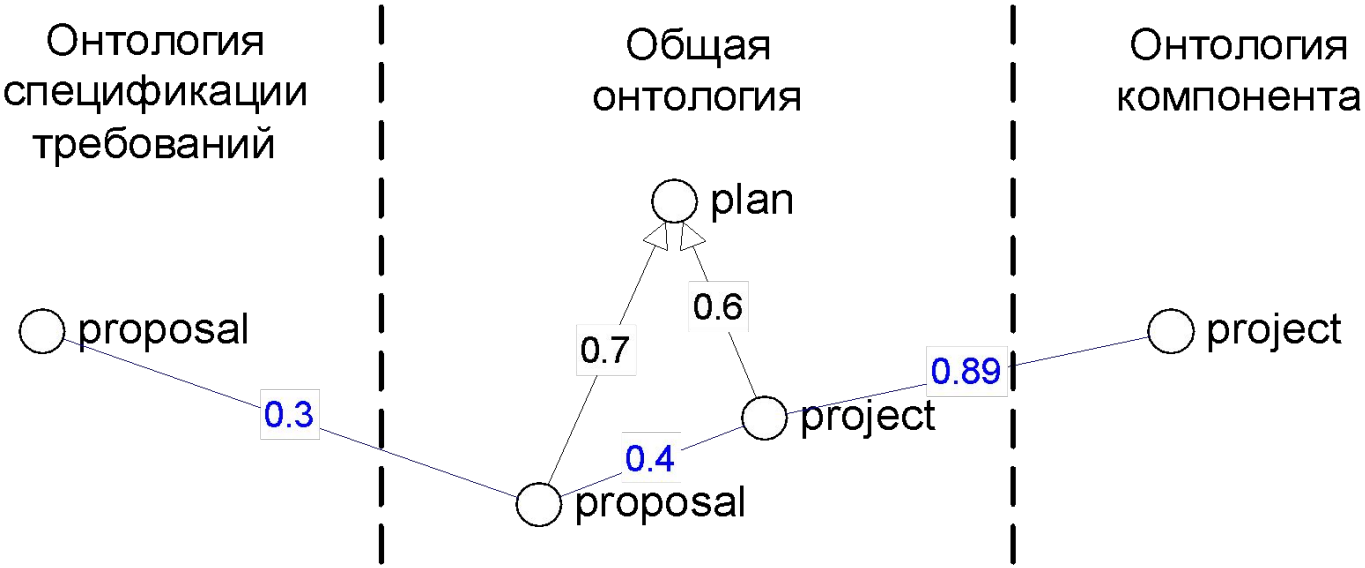
Поиск онтологически релевантных элементов

- Онтологическая модель служит основой для поиска спецификаций релевантных информационных компонентов и их фрагментов
- Нужен алгоритм отображения онтологических понятий спецификации требований (и компонентов) в понятия общей онтологии
- Требуется алгоритм установления ассоциаций между понятиями спецификации требований и компонентов на основе композиции ассоциаций между понятиями
- Нужен алгоритм установления ассоциаций между элементами схем спецификации требований и компонентов

Онтологическая модель

- С каждым элементом спецификации требований и компонентов связывается соответствующее онтологическое понятие (аннотация элемента)
- Спецификация онтологического понятия содержит:
  - описание на естественном языке
  - ассоциации обобщения/специализации
  - позитивные ассоциации
- Использование близких по смыслу онтологических спецификаций в разных компонентах является необходимой предпосылкой корректной взаимной интерпретации элементов спецификаций

Установления ассоциаций между онтологическими понятиями



- Функция позитивной корреляции
  - $sim(X,Y) = (Cx,Cy) / |Cx|*|Cy|$
- Функции вычисления ассоциаций обобщения/специализации
  - $r(X,Y) = \Sigma_i(min(Cx_i,Cy_i)) / |Cx|$
  - $r(Y,X) = \Sigma_i(min(Cx_i,Cy_i)) / |Cy|$

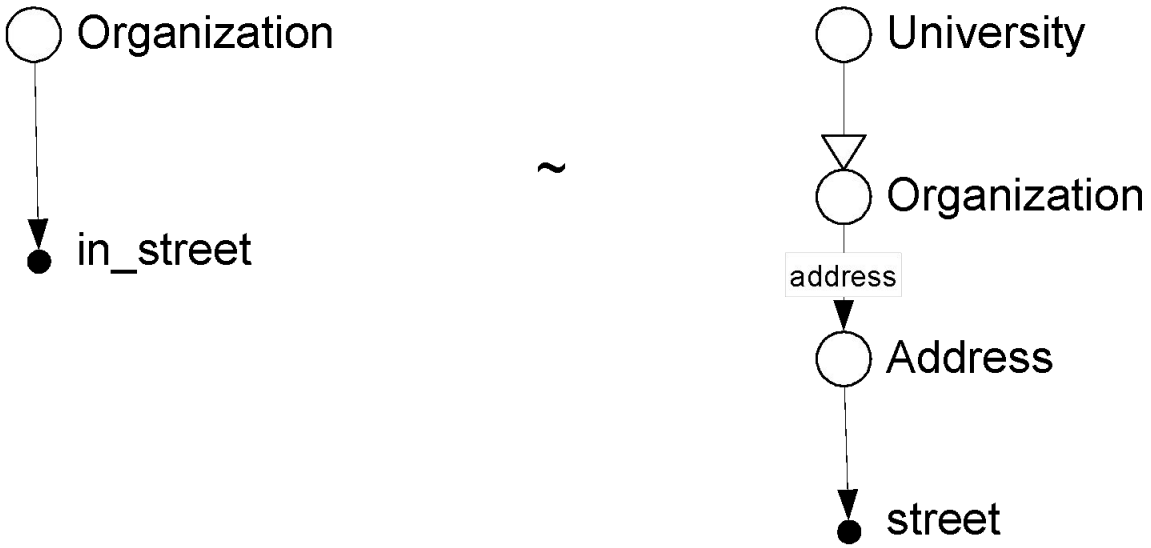
Разрешение конфликтов между спецификациями

- Необходим метод разрешения различных рассогласований между спецификациями требований и компонентов
- Набор правил структурных преобразований между спецификациями требований и компонентов
- Алгоритм автоматического разрешения структурных конфликтов

Метод разрешения конфликтов

- Виды конфликтов
  - структурные конфликты
  - конфликты значений
  - конфликты поведения
- Подходы к разрешению конфликтов
  - применение языка высокого уровня для описания функций разрешения конфликтов
  - применение набора предопределенных правил структурных преобразований

Пример разрешения структурного конфликта




```
get_in_street: {in: function;
  params: {+y/University,
           -returns/Organization.in_street};
  {{ returns = y.address.street }}
}
```

Пример правила структурных преобразований

- Условия потенциальной релевантности путей
  - полнота
  - ацикличность
  - минимальность
- Путь *X0...X1* релевантен пути *Y0...Y1*, если они удовлетворяют условию потенциальной релевантности и имеют следующий вид:
  - $X0–a1→X1 \sim Y0(–b→| \Rightarrow)^*Y3–b1→Y1$
- При этом автоматически конструируемая функция разрешения конфликтов имеет следующий вид:

```
– get_a1: {in: function;
    params: {+y/Y0, -returns/X1};
    {{ returns = y(.b)*.b1 }}}
```

Конструирование композиций компонентов

- Конструирование композиции фрагментов спецификации компонентов с использованием операций над типами
- Алгоритм конструирования наибольшего общего редукта
- Алгоритмы реализации операций над типами: объединения (join) и пересечения (meet) двух спецификаций типов-операндов
- Алгоритм конструирования уточняющих композиций спецификаций как программных, так и информационных компонентов

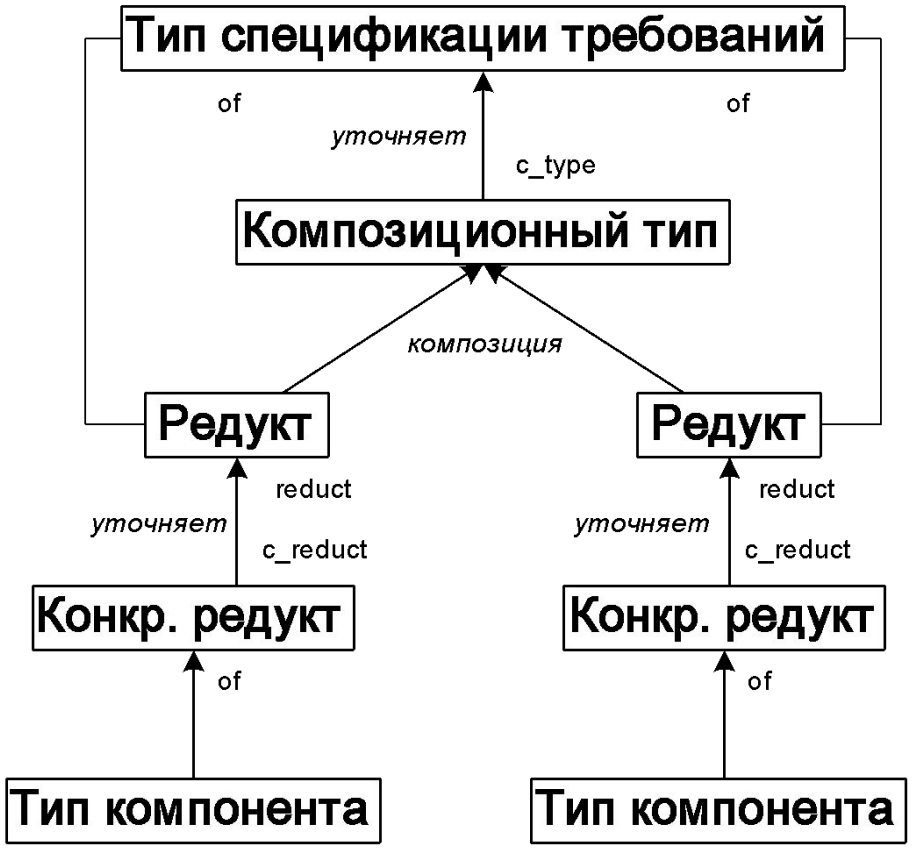
Подход к конструированию композиций компонентов

- Процесс конструирования основан на *выявлении фрагментов спецификаций* существующих компонентов и их дальнейшей *композиции*, уточняющей спецификацию требований.
- Для этого используются операции над типами, ведущими к трансформации их спецификаций – операции *декомпозиции* и *композиции*.
- *Уточняющие спецификации*, образуемые при конструировании, согласно теории уточнения, могут использоваться всюду *вместо уточняемых* спецификаций требований, так что пользователи не замечают этой замены.

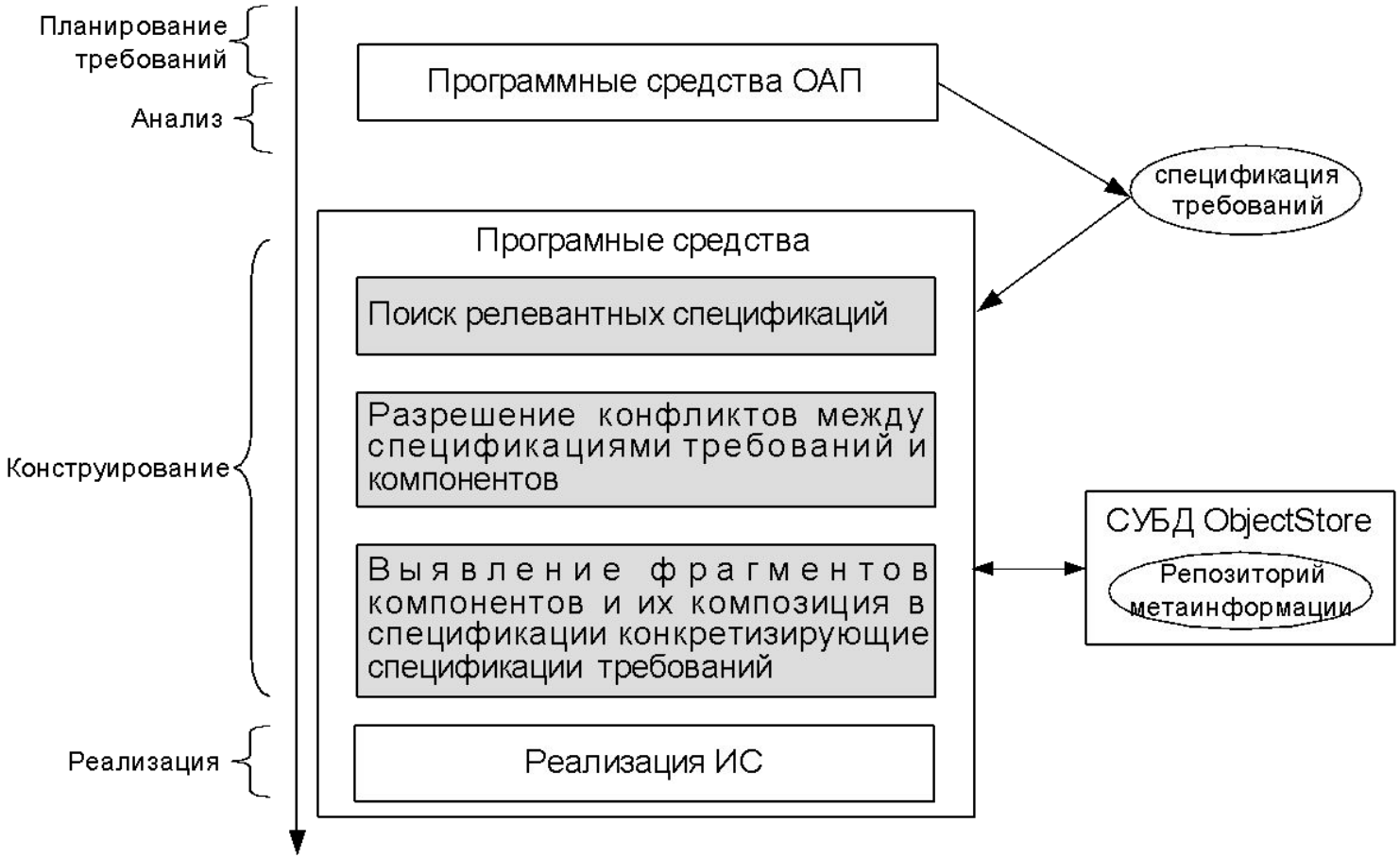
Операции над спецификациями типов

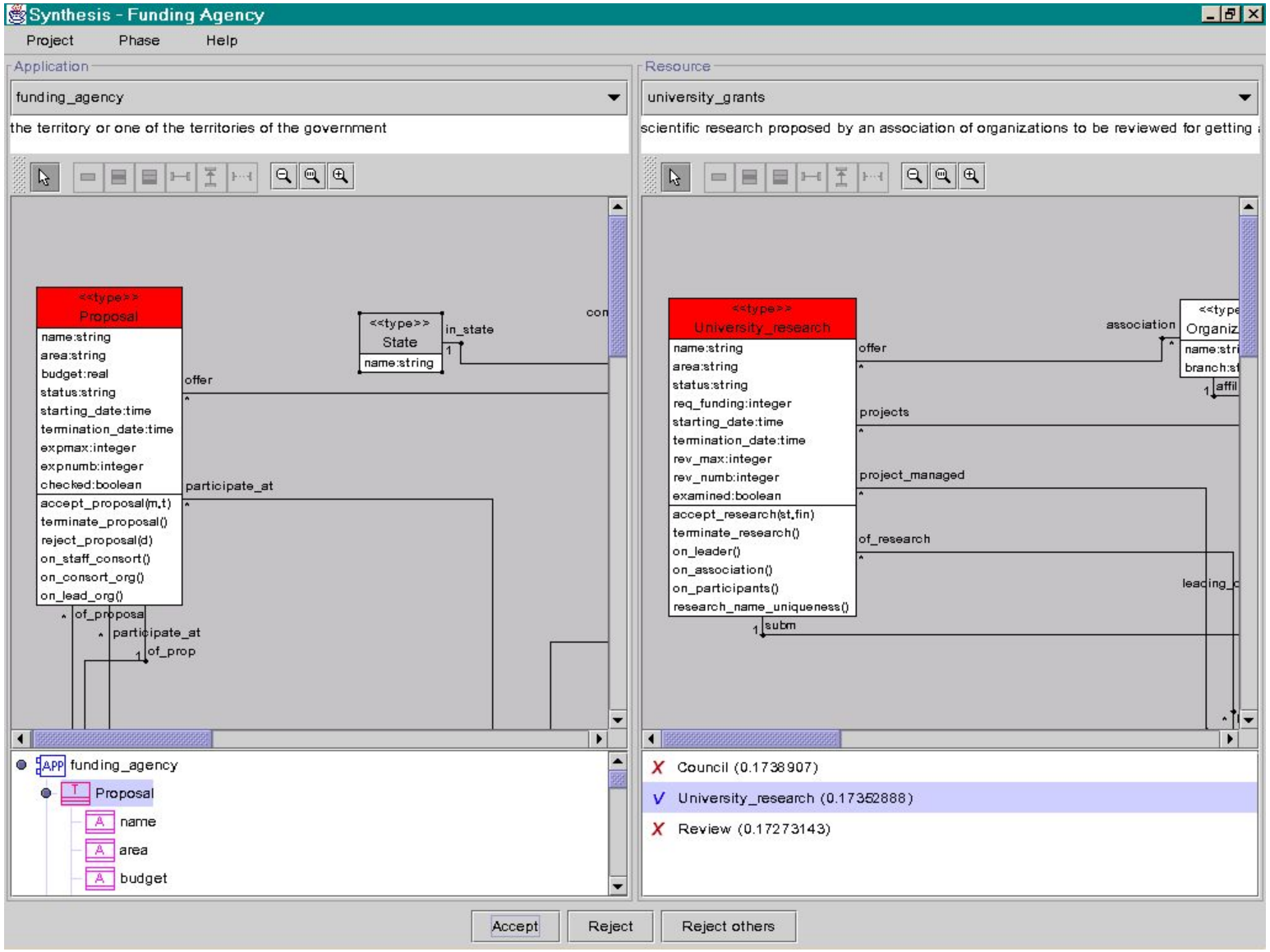
- *Редукт* R_T типа *T* определяется как подсигнатура сигнатуры типа, при этом множество атрибутов редукта является подмножеством множества атрибутов типа, множество функций редукта является подмножеством множества функций типа, множество предикатов редукта является подмножеством множества предикатов типа.
- *Общий редукт* для типов *T1*, *T2* есть редукт R_{T1} типа *T1* такой, что существует редукт R_{T2} типа *T2*, при этом редукт R_{T2} является уточнением редукта R_{T1}
- *Наибольший общий редукт* $R_{MC}(T1,T2)$
- Операция *meet*. Операция $T1 \sqcap T2$ образует тип T как пересечение спецификаций типов-операндов. Тип T образуется слиянием двух наиболее общих редуктов типов T1 и T2 : $R_{MC}(T1,T2)$ и $R_{MC}(T2,T1)$ – включающее в себя объединение множеств их операций и дизъюнкцию инвариантов.
- Операция *join*

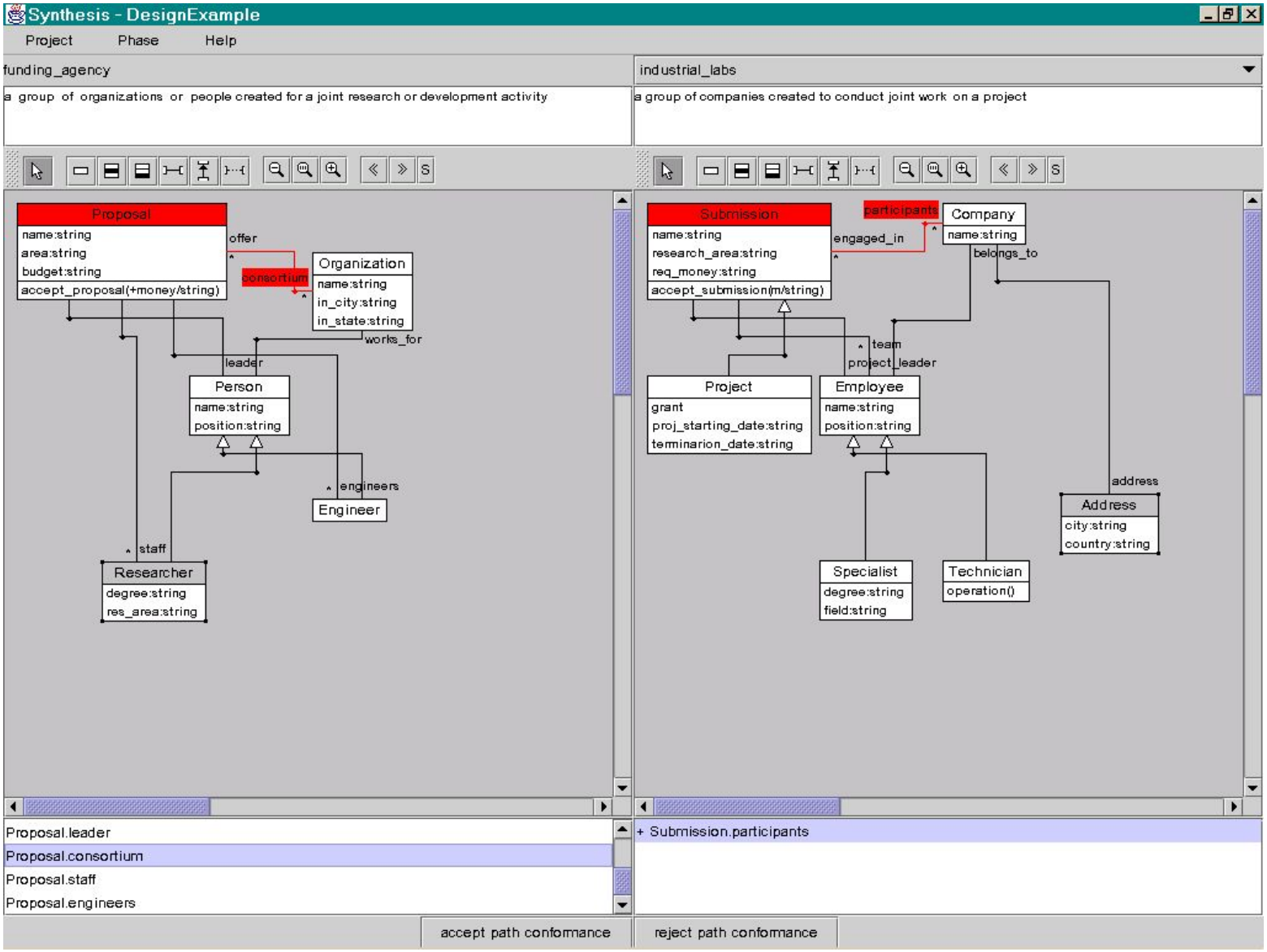
Иерархия спецификаций композиционных типов

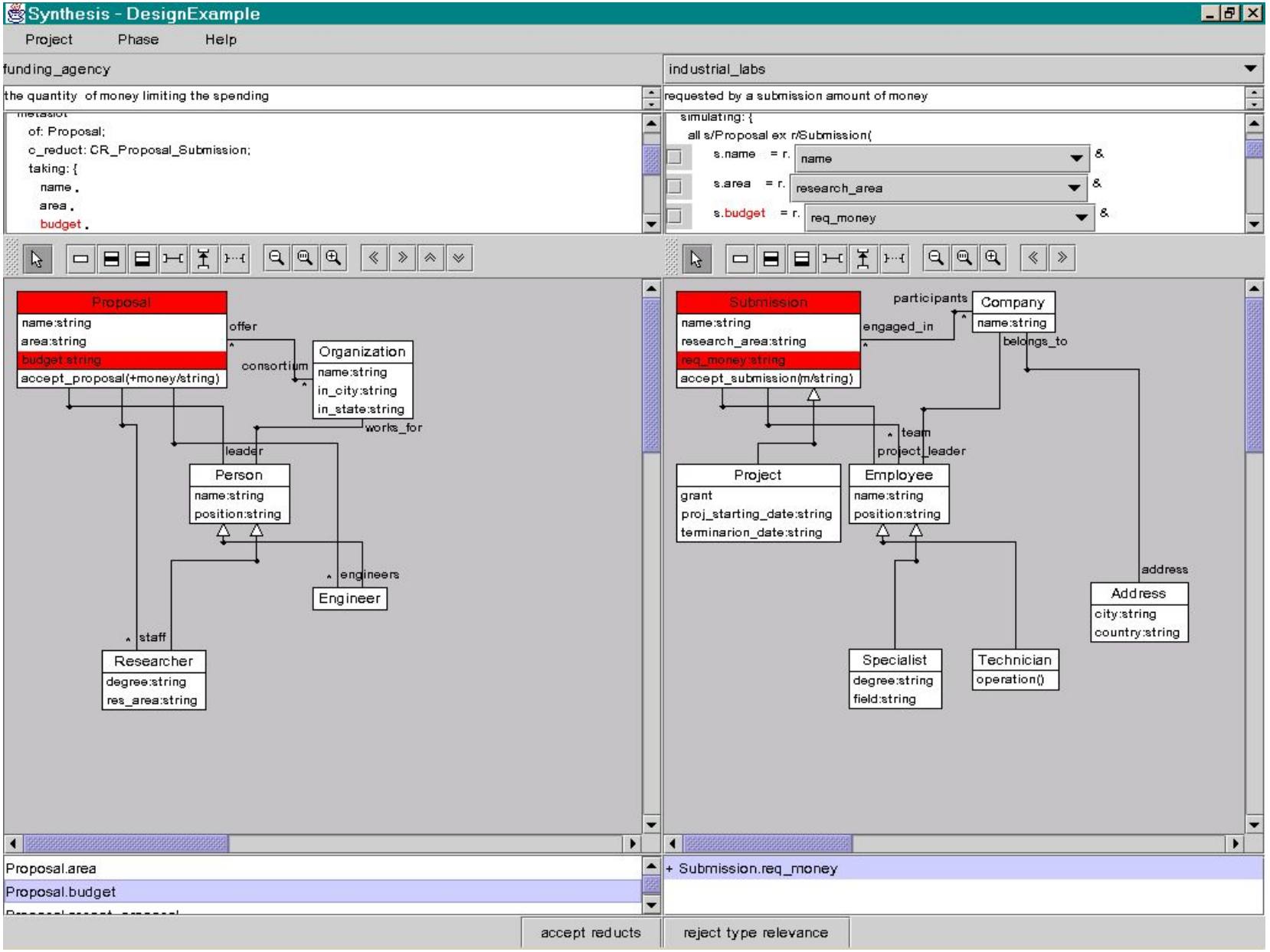


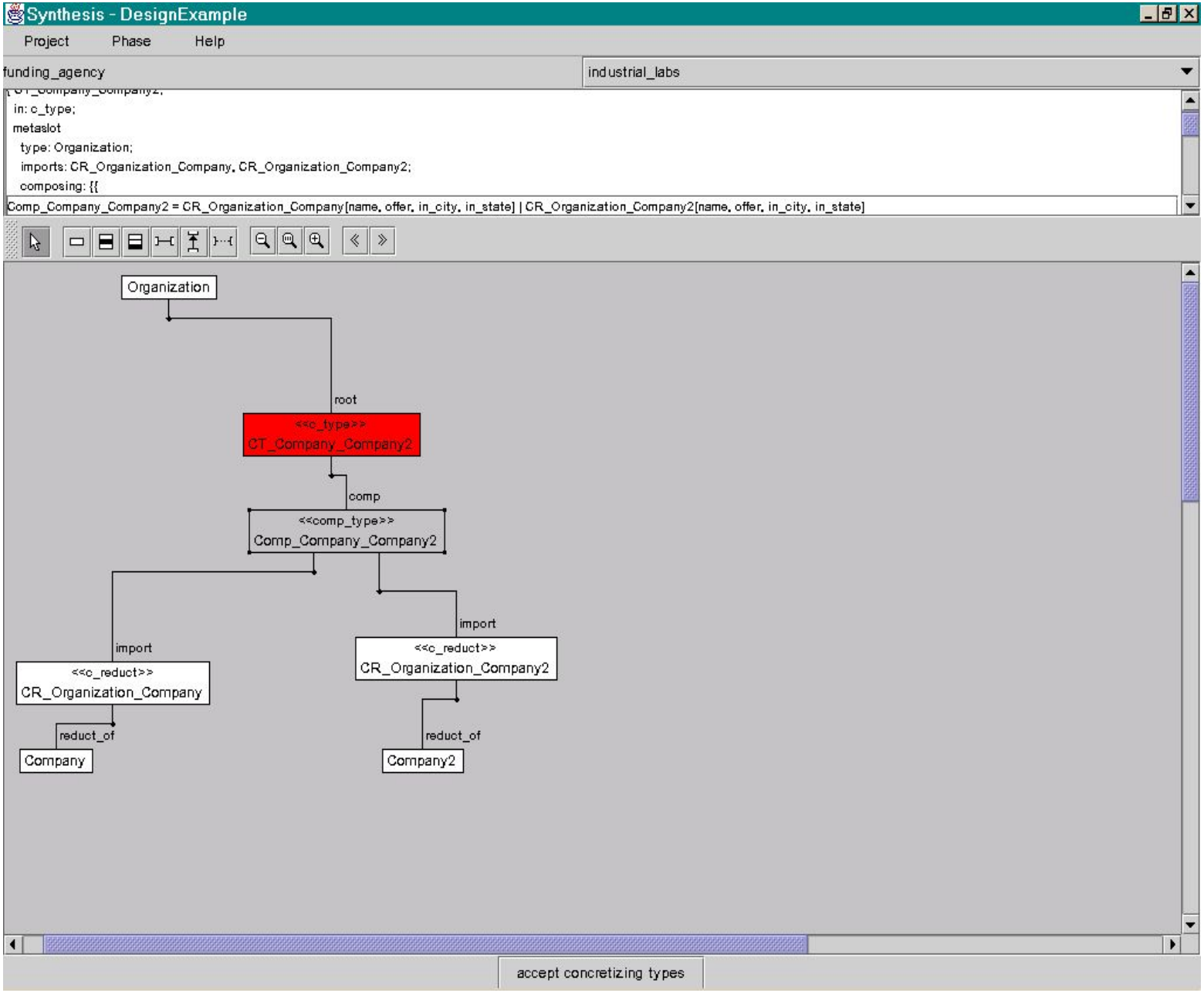
Архитектура инструментария эксперта-конструктора ИС




Synthesis - Funding Agency
Project
Phase
Help
Application
funding_agency
the territory or one of the territories of the government
Resource
university_grants
scientific research proposed by an association of organizations to be reviewed for getting
<<type>>
Proposal
name:string
area:string
budget:real
status:string
starting_date:time
termination_date:time
expmax:integer
expnumb:integer
checked:boolean
accept_proposal(m,t)
terminate_proposal()
reject_proposal(d)
on_staff_consort()
on_consort_org()
on_lead_org()
<<type>>
State
name:string
in_state
offer
participate_at
of_proposal
participate_at
of_prop
<<type>>
University_research
name:string
area:string
status:string
req_funding:integer
starting_date:time
termination_date:time
rev_max:integer
rev_numb:integer
examined:boolean
accept_research(st,fin)
terminate_research()
on_leader()
on_association()
on_participants()
research_name_uniqueness()
offer
association
projects
project_managed
of_research
subm
funding_agency
Proposal
name
area
budget
Council (0.1738907)
University_research (0.17352888)
Review (0.17273143)
Accept
Reject
Reject others


Synthesis - DesignExample
Project
Phase
Help
funding_agency
a group of organizations or people created for a joint research or development activity
industrial_labs
a group of companies created to conduct joint work on a project
Proposal
name:string
area:string
budget:string
accept_proposal(+money/string)
offer
consortium
Organization
name:string
in_city:string
in_state:string
works_for
leader
Person
name:string
position:string
engineers
Engineer
staff
Researcher
degree:string
res_area:string
Submission
name:string
research_area:string
req_money:string
accept_submission(m/string)
participants
engaged_in
Company
name:string
belongs_to
team
project_leader
Project
grant
proj_starting_date:string
terminarion_date:string
Employee
name:string
position:string
address
Address
city:string
country:string
Specialist
degree:string
field:string
Technician
operation()
Proposal.leader
Proposal.consortium
Proposal.staff
Proposal.engineers
+ Submission.participants
accept path conformance
reject path conformance


Synthesis - DesignExample
Project
Phase
Help
funding_agency
industrial_labs
the quantity of money limiting the spending
requested by a submission amount of money
of: Proposal;
c_reduct: CR_Proposal_Submission;
taking: {
name .
area .
budget .
simulating: {
all s/Proposal ex r/Submission(
s.name = r. name &
s.area = r. research_area &
s.budget = r. req_money &
Proposal
name:string
area:string
budget:string
accept_proposal(+money/string)
offer
consortium
Organization
name:string
in_city:string
in_state:string
works_for
leader
Person
name:string
position:string
engineers
Engineer
staff
Researcher
degree:string
res_area:string
Submission
name:string
research_area:string
req_money:string
accept_submission(m/string)
participants
Company
name:string
engaged_in
belongs_to
team
project_leader
Project
grant
proj_starting_date:string
terminarion_date:string
Employee
name:string
position:string
address
Address
city:string
country:string
Specialist
degree:string
field:string
Technician
operation()
Proposal.area
Proposal.budget
+ Submission.req_money
accept reducts
reject type relevance


Synthesis - DesignExample
Project
Phase
Help
funding_agency
industrial_labs
in: c_type;
metaslot
type: Organization;
imports: CR_Organization_Company, CR_Organization_Company2;
composing: {{
Comp_Company_Company2 = CR_Organization_Company[name, offer, in_city, in_state] | CR_Organization_Company2[name, offer, in_city, in_state]
Organization
root
<<c_type>>
CT_Company_Company2
comp
<<comp_type>>
Comp_Company_Company2
import
<<c_reduct>>
CR_Organization_Company
reduct_of
Company
import
<<c_reduct>>
CR_Organization_Company2
reduct_of
Company2
accept concretizing types